Precision Farming

Muhammad Umer Bin Yaqoob, Ammar Khan, Jiban-Ul Azam Chowdhary Shafin[1]

Contents

1

Abstract: Precision farming has been instrumental in revolutionising modern agriculture by enabling the precise application of agricultural inputs such as chemicals, seeds, and fertilisers, thereby ensuring cost-effective and sustainable production. Farmers can gain deeper insights into field conditions by leveraging advanced technologies like drones, machine learning models, and high-resolution mapping systems, facilitating timely and precise interventions.This documentation explores the integration of precision farming techniques for crop classification, weed detection, and real-time problem identification. The study underscores the pivotal role of cutting-edge tools in optimising resource utilisation, minimising costs and enhancing yields.The objective is to demonstrate the effective implementation of innovative technologies such as drones and advanced machine-learning techniques in real-world agricultural practices, thereby improving productivity and profitability while supporting sustainable farming methods.

1 Introduction And Motivation

The world's population is expected to grow by 2 billion by 2050, according to the United Nations [UNSA19]. To meet the growing demand for food, farmers need to maximize production. The introduction of precision farming is seen as the way forward. Precision farming improves efficiency, increases productivity, and reduces costs, thus providing a sustainable way to combat hunger ensuring food security.

According to Bahr et al. [Ba15], precision farming is a system that emphasizes real-time monitoring of plants, animals, and fields and provides targeted responses. This approach often involves robotic technologies, such as unmanned aerial vehicles (UAVs), which are used to continuously monitor and respond to the specific needs of crops or agricultural areas [Ra19]. By optimising resource use, minimising labour costs and increasing crop yields, these practices significantly improve agricultural outcomes.

In this project, we use the concept of cyber-physical systems (CPS) together with deep learning algorithms to design and implement a precision farming prototype. We start by analysing the system and proposing an architecture. This is followed by illustrating the concept through use case diagrams. We then model the design using Petri nets supported by TAPAAL software. In addition, we use fastai, a deep learning framework based on PyTorch and Fastai, to develop and train our algorithm. Our main focus in this project is the detection and removal of weeds and watering of plants.

1.1 Foundation

Precision Farming involves using the minimum amount of resources and getting the maximum yield by leveraging modern technologies like drones, sate-light imaging and advance learning algorithms. We have identified several use-cases.The central idea of this approach is to use a monitoring drone to take periodic aerial images or sensor readings of the farmland. These images are then fed into a machine learning model - such as a ResNet or similar convolutional neural network - that has been trained to distinguish between healthy crops and various types of unwanted plants or pests. If a weed is detected or an area appears to be stressed, the system flags that spot as needing intervention. This classification step, enhanced by data augmentation and careful labelling, is critical to achieving reliable performance in the field. Instead of spraying herbicides or pesticides across an entire field, the system pinpoints exactly where control measures are needed, reducing chemical use and minimising damage to surrounding vegetation and soil micro-organisms. In addition, sensors (such as infrared or multi-spectral cameras) can provide details of plant health indicators such as chlorophyll levels, helping to detect disease at an early stage.

2 Use Case

As illustrated in Figure 1, entitled "Precision Farming Drone Communication System", three primary entities collaborate within a cohesive operational environment: the monitoring drone (UAV), the weed/pest control drone, and the irrigation drone.The monitoring drone is responsible for conducting field monitoring, encompassing aerial surveillance of the farmland and the measurement of soil moisture using IR sensors. In the event of the monitoring drone detecting weeds or pests, as depicted by the Weed Detected and Pest Detected use cases, respectively, it initiates the process of transmitting coordinates to the relevant drone. This action is denoted by the «include> relationships, which indicate that the subtasks (i.e. Destroy Weed or Destroy Pest) depend on information gathered during field monitoring. Consequently, the Weed/Pest Destroying Drone is able to accurately target problem areas and carry out weed or pest control operations with minimal waste or ecological disruption.Similarly, if the drone's IR sensors detect insufficient soil moisture (as captured by the 'Low Water Detected IR Sensors' use case), coordinates are transmitted to the Watering Drone, which performs the 'Water the Target Field' operation. The integrated workflow, as depicted in Figure 1, underscores the necessity for seamless real-time monitoring and task execution in precision farming systems.

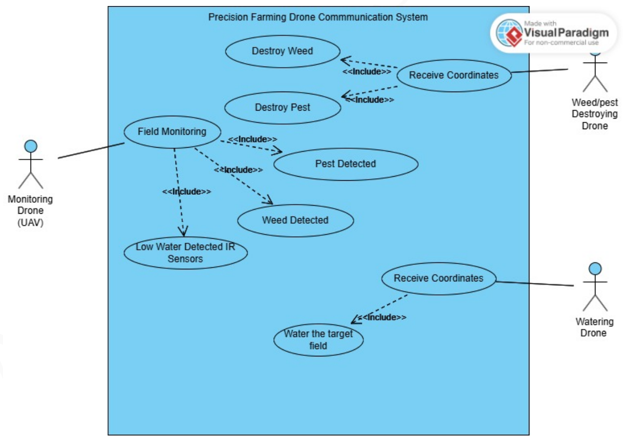


Fig. 1: Precision Farming Drone Communication System

As illustrated in Figure 2, which is labelled Överall System", a more comprehensive perspective on the data flow and system orchestration becomes apparent.The Monitoring Drone is responsible for acquiring information pertinent to analysing plant health, monitoring water levels and detecting pest presence. It then transmits these findings via the Data Transfer 1 pathway to the System Controller. The System Controller then dispatches commands to various specialised components, including an Autonomous Vehicle, a Weeding Drone, and a Watering Drone, each responsible for discrete operational tasks.The Autonomous Vehicle may handle navigation-related functions (e.g. Avoid Obstacles for Safety), while the Weeding Drone and Watering Drone execute actions such as Apply Pesticides and

Water Plants, respectively. A secondary channel, designated Data Transfer 2, underscores the system's capacity to relay instructions or updates back to the drones.This architectural approach enables a multitude of additional use cases, including the capacity to assess the readiness of plants for harvesting. This suggests that the communication framework can support a diverse array of agricultural procedures. As illustrated by Figure 2, a scalable and modular design is depicted, in which precision farming operations—ranging from basic crop surveillance to advanced interventions—are seamlessly integrated.

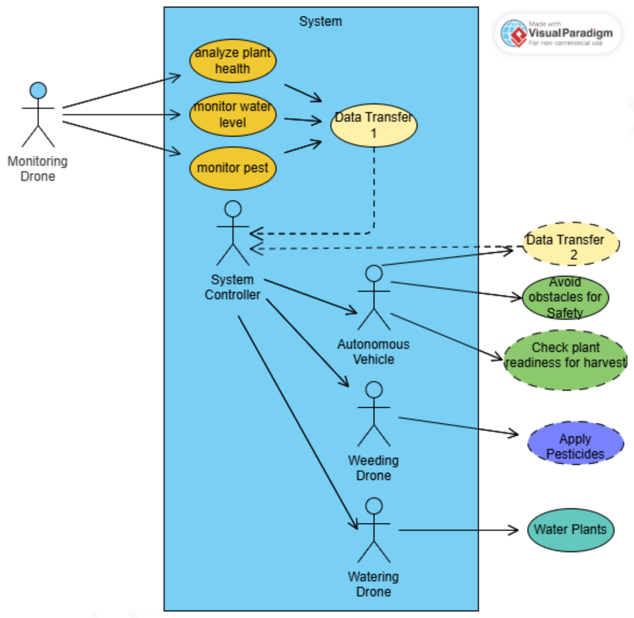


Fig. 2: Overall System

3 Modelling and Designing

In order to ensure a rigorous and verifiable workflow for precision farming tasks, TAPAAL (Timed-Arc Petri Net Analyzer) was employed to model the key operational aspects of the system. TAPAAL provides both a simulation and analysis environment that enables the specification of timing constraints, resource allocation, and interaction sequences among the drones. The Petri net models encapsulate the logic of deployment, task execution, and system reset, thereby ensuring that each stage of the farming process – monitoring, weeding, watering, and autonomous navigation – is both transparent and formally verifiable.

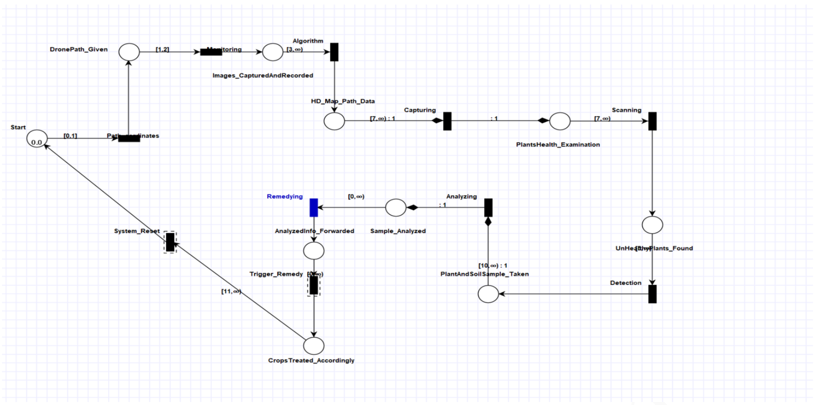


Fig. 3: Precision Farming Drone Communication System

In order to capture the operational flow of drones within a precision farming context, a series of timed-arc Petri net models were constructed using TAPAAL. These models depict crucial system states, transitions, and synchronization mechanisms that ensure coherent task execution among various drones, namely the Monitoring Drone, the Weeding Drone, and the Watering Drone, as well as an autonomous vehicle.

The initial model delineates the overarching workflow, commencing at system start-up and proceeding through path assignment, data capture, analysis, and eventual system reset.Subsequent diagrams highlight the coordination process, wherein a central trigger orchestrates remedy actions – such as weed removal or irrigation – by engaging respective drones.Each drone model incorporates path-allocation transitions, timed invariants to bound operational periods, and completion states indicating readiness for new assignments.

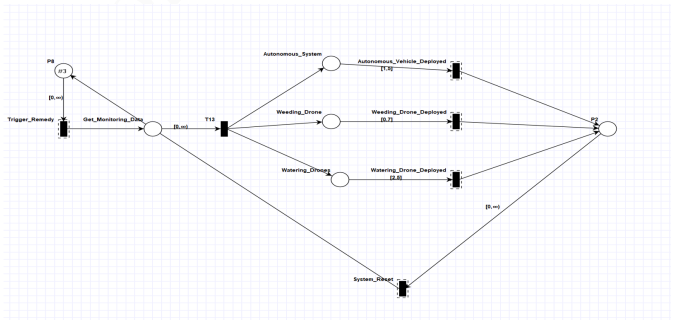
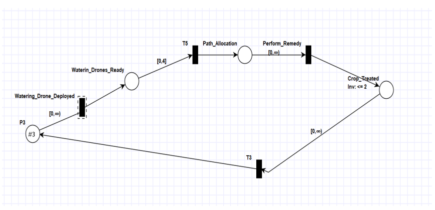
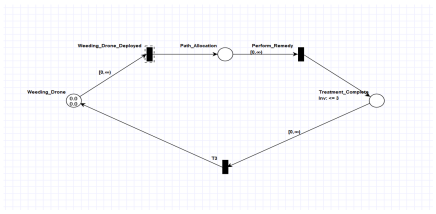


Fig. 4: Controller in Drone Communication System

In the controller model, the tokens are channelled to the autonomous vehicle, the weeding drone, or the watering drone. Each drone's model features path-allocation transitions, where routes are assigned according to detected anomalies, and a timed "perform remedy" operation that must complete within a specified interval.



(a) Watering Drone System



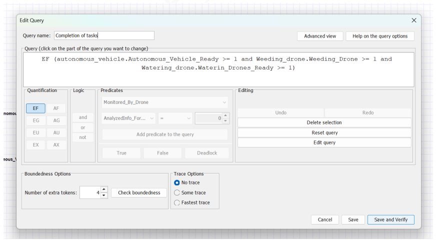
(b) Weeding Drone System

Each drone model is equipped with path-allocation transitions, wherein routes are assigned in accordance with detected anomalies, and a timed "perform remedyöperation that must be completed within a specified interval. For the weeding drone, an invariant caps treatment durations to ensure rapid turnover, whereas for the watering drone, another timed limit ensures efficient irrigation and drone availability for subsequent tasks.

Collectively, these Petri nets provide a structural approach to verifying the correctness of multi-drone interactions. They facilitate the detection of potential conflicts (e.g., simultaneous deployments in the same area) and ensure logical consistency (e.g., avoiding watering before weeds are removed).

3.1 TAPAAL Queries

To further validate the correctness of the system and to ensure that critical tasks can actually be completed, we used TAPAAL's model checking functionality by formulating queries in a temporal logic style. One such query, labelled "Completion of tasks", is shown in the figure a and b.



Query name: Completion of tasks

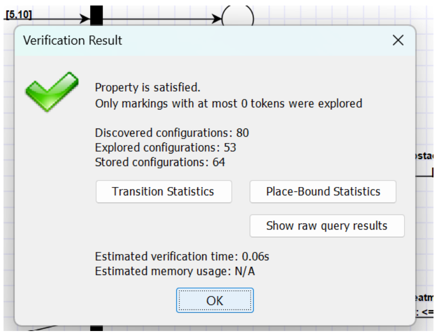
Query (click on the part of the query you want to change)

```
EF (autonomous_vehicle.Autonomous_Vehicle_Ready >= 1 and Weeding_drone.Weeding_Drone >= 1 and
Watering_drone.Waterin_Drones_Ready >= 1)
```

Boundedness Options
Number of extra tokens: 4

Trace Options
[✓ No trace] Some trace Fastest trace

(a) Queries



Property is satisfied.
Only markings with at most 0 tokens were explored

Discovered configurations: 80
Explored configurations: 53
Stored configurations: 64

Estimated verification time: 0.06s
Estimated memory usage: N/A

(b) Weeding Drone System

In simple terms, the EF operator in TAPAAL checks whether it's possible for all the drones the autonomous vehicle, the weeding drone and the irrigation drone - to reach a 'ready' state, where they can perform their tasks. This check confirms that the farming process can proceed as planned, despite timing and coordination constraints.

During the verification, TAPAAL confirmed that the system was working as intended. It found an achievable state where all drones are ready, showing that the workflow is achievable without problems such as deadlocks or missing steps. The tool also provided useful metrics to validate the reliability and efficiency of the system.

4 Deep Learning Techniques for Optimizing Precision Farming

4.1 Dataset

The dataset used for training was provided by our professor and consisted of the following plant classes as shown in the Figure 6:

- Shepherd's Purse
- Common Wheat
- Fat Hen
- Maize
- Common Chickweed
- Cleavers
- Sugar Beet
- Small-flowered Cranesbill
- Scentless Mayweed
- Loose Silky-bent
- Black-grass
- Charlock

4.2 Data Preparation

We used Fastai's as shown in the Figure 7 snippet `ImageDataLoaders.from_folder` method to load and transform the dataset. The training-validation split was set to 80%-20% (`valid_pct=0.2`), and a seed (`seed=42`) was defined for reproducibility. Each image was resized to 224x224 pixels (`Resize(224)`) to match the ResNet50 input requirements. To improve model generalization, we applied augmentation transformations such as flipping, rotating up to 30 degrees, and zooming (`aug_transforms`) as shown in Figure 8.

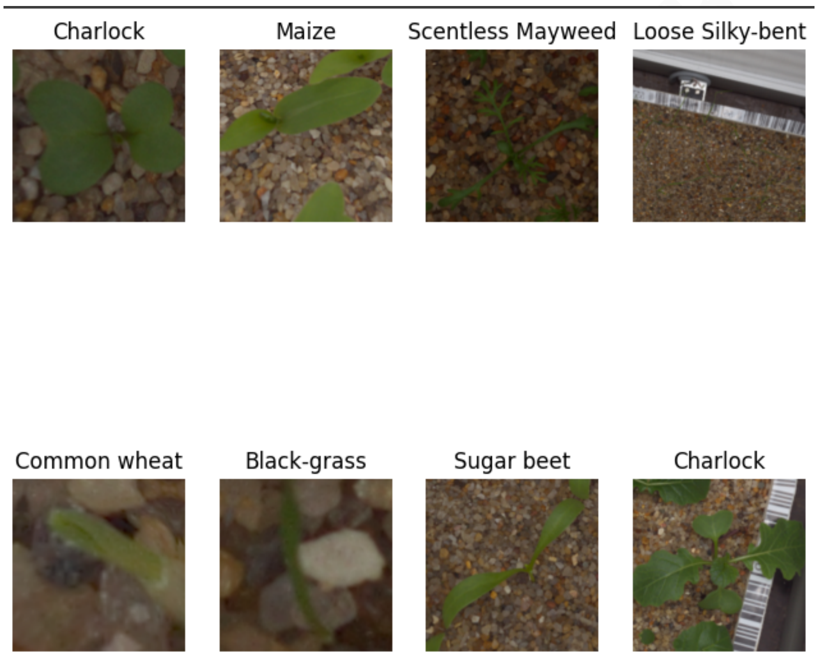
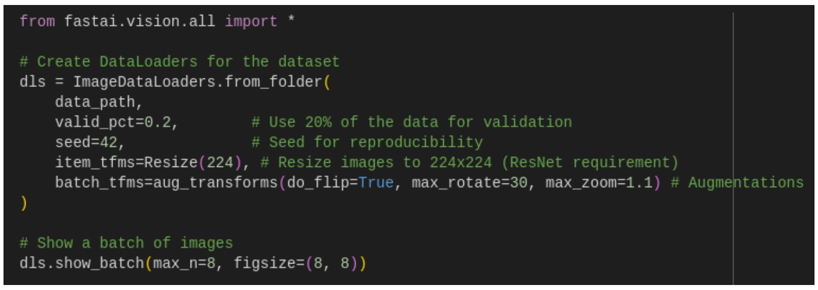


Fig. 7: Plants Classes



```
from fastai.vision.all import *

# Create DataLoaders for the dataset
dls = ImageDataLoaders.from_folder(
    data_path,
    valid_pct=0.2,        # Use 20% of the data for validation
    seed=42,              # Seed for reproducibility
    item_tfms=Resize(224), # Resize images to 224x224 (ResNet requirement)
    batch_tfms=aug_transforms(do_flip=True, max_rotate=30, max_zoom=1.1) # Augmentations
)

# Show a batch of images
dls.show_batch(max_n=8, figsize=(8, 8))
```

Fig. 8: Data Loader

4.3 Model Selection

We selected the ResNet50 architecture as our model backbone due to its deep residual learning capabilities [He16]. ResNet50 is widely used in image classification tasks and effectively mitigates vanishing gradient issues, making it a robust choice for our application.

4.4 Finding the Optimal Learning Rate with Fastai

To train such a deep learning model, we first create a learner using ResNet50 as the backbone. We use `vision_learner` method from Fastai to define the model with accuracy as the metric:

```
# Create a learner with ResNet50
learn_plant = vision_learner(
    dls,
    resnet50,  # Use ResNet50 as the backbone
    metrics=accuracy  # Track accuracy during training
)
```

Then, we apply the `lr_find()` function in order to find the best learning rate for training. It performs a learning rate range test and plots the loss curve against different learning rates:

```
# Find the optimal learning rate
learn_plant.lr_find()
```

The output of this command downloads the ResNet50 pretrained weights if not already available and generates a plot of the loss vs. learning rate. The plot suggests an optimal learning rate where the loss decreases steeply before increasing again.

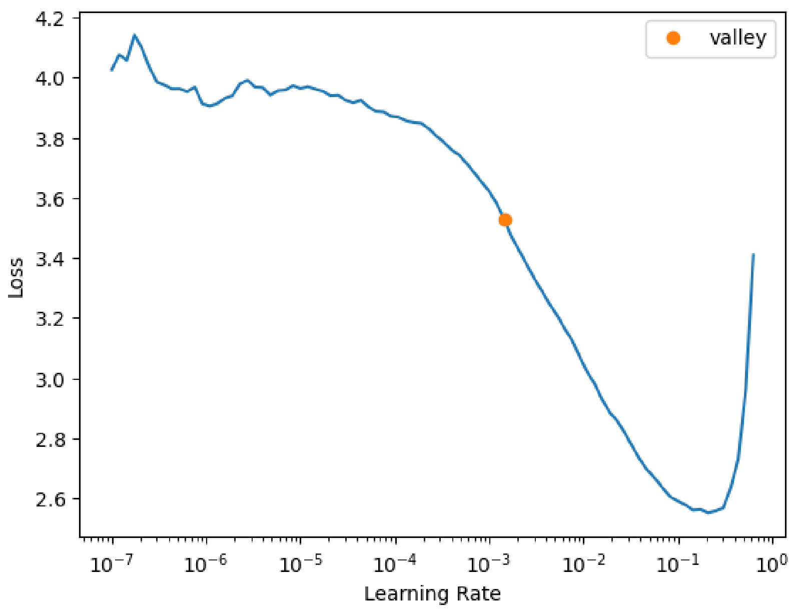


Fig. 9: Optimal Learning Rate Selection

In the plot shown in Figure 9, the valley point (highlighted in orange) represents the suggested learning rate for optimal training. The suggested learning rate is approximately 0.00144, which is the point where the loss is decreasing the fastest before flattening out.

We employed Fastai's learning rate finder (`learn_plant.lr_find()`) to determine an optimal learning rate for efficient convergence [Sm17].

4.5 Fine-tuning the Model with Fastai

After identifying the optimal learning rate (`lr_find()`), the model was fine-tuned using the `fine_tune()` method. The following code snippet demonstrates the process:

```
# Fine-tune the model
learn_plant.fine_tune(5, base_lr=0.001)
```

The `fine_tune()` function trains the model in two phases:

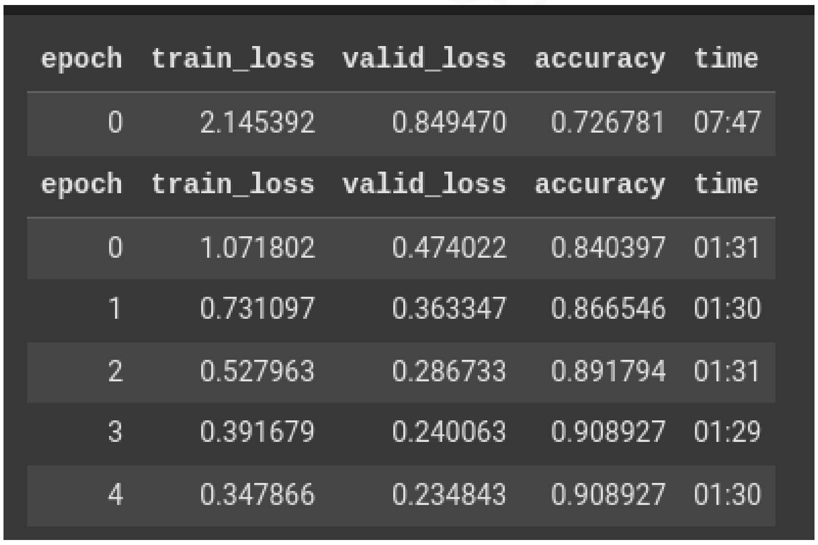
4.6 First Phase (Epoch 0)

During the first phase, the pretrained ResNet50 backbone is frozen, and only the newly added layers are trained. This allows the model to adapt to the new dataset without overwriting the general features learned by ResNet50.

- **Train Loss:** 2.145392
- **Validation Loss:** 0.849470
- **Accuracy:** 72.68%

4.7 Second Phase (Epochs 1-4)

In the second phase shown in Figure 10, the entire model, including the pretrained backbone, is unfrozen and fine-tuned. A learning rate of 0.001 is used to ensure a gradual adjustment of all layers, improving both validation loss and accuracy.



epoch	train_loss	valid_loss	accuracy	time
0	2.145392	0.849470	0.726781	07:47

epoch	train_loss	valid_loss	accuracy	time
0	1.071802	0.474022	0.840397	01:31
1	0.731097	0.363347	0.866546	01:30
2	0.527963	0.286733	0.891794	01:31
3	0.391679	0.240063	0.908927	01:29
4	0.347866	0.234843	0.908927	01:30

Fig. 10: Fine-tuning results with Fastai

4.8 Confusion Matrix Analysis

To evaluate the classification performance of the model, a confusion matrix is generated as shown in Figure 11. The confusion matrix provides insights into the model's predictions by showing how many samples from each true class were correctly or incorrectly classified.

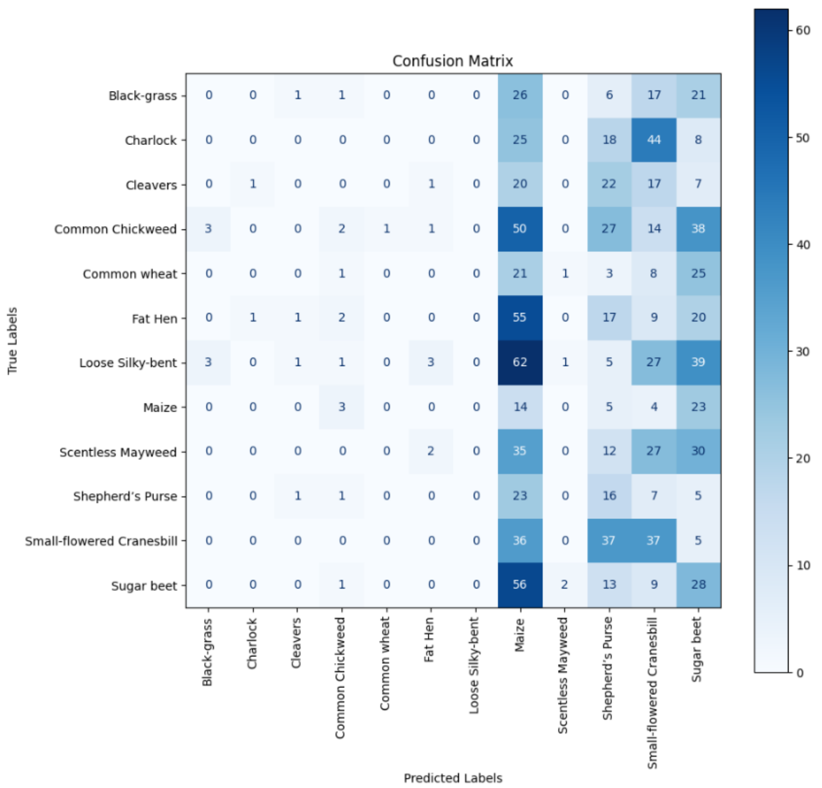


Fig. 11: Confusion Matrix for Model Predictions

The confusion matrix helps in understanding how well the model distinguishes between different classes:

- **Diagonal Elements:** These represent correctly classified instances. Higher values along the diagonal indicate better classification performance.
- **Off-Diagonal Elements:** These represent misclassified instances, showing which classes are confused with each other.
- **Class-Specific Errors:** Some categories, such as *Loose Silky-bent* and *Sugar Beet*, have higher misclassification rates, indicating that these classes are harder to differentiate.

By analyzing this confusion matrix, we can refine our model by applying additional data augmentation techniques, adjusting the learning rate, or using a more complex architecture to further improve classification accuracy.

The matrix shows a reasonable spread of correct predictions (high numbers on the diagonal) for several classes, but there are also notable misclassifications. Some classes like "Common Chickweed“, and “Loose Silky-ben"have higher misclassifications compared to others, indicating potential confusion or feature similarity between classes.

the model performs reasonably well overall but could benefit from further tuning and training, particularly to handle classes that are currently prone to higher misclassification rates.

4.9 Recommendations for Improvement:

Perform various data augmentation techniques to increase diversity and quantity in training data, especially within the badly performing classes. Feature Engineering: Enhance feature extraction processes or try different model architectures that could capture the subtlety between classes better. Hyperparameter Tuning: Try different learning rates, batch sizes, or other relevant hyperparameters. Employ ensemble methods or transfer learning; the most advanced architectures for the same are EfficientNet and Transformers for image classification. It is nice to see the overall classes have a reasonable degree of accuracy, but this should be enhanced by reducing misclassifications.

5 Model Verification and Performance Evaluation

After training the model, we performed an extensive verification process to evaluate its performance. We selected a subset of images from the dataset, ranging from image indices 000 to 899, and used them as test samples. This step ensured that our model generalized well beyond the training data.

5.1 Cross-verification with Actual Labels

To validate the results, we cross-verified the model's predictions with the actual data. The verification dataset contained images along with their class labels, allowing us to compare the predicted output against the correct classification.

- **Actual Classes:** The model classified images and produced outputs similar to the following format:

Image Index	Actual Classes
56	Black-grass
57	Black-grass
58	Black-grass
59	Black-grass
60	Black-grass

Tab. 1: Actual Data Classification

- **Predicted Classification :** The Predicted classification data, against which actual data were verified, was structured as follows:

Image Index	Predicted Classes
000	Black-grass
001	Loose Silky-bent
003	Black-grass
004	Black-grass
005	Loose Silky-bent

Tab. 2: Model Prediction

5.2 Analysis of Model Performance

Therefore, by comparing these results, we ensured that the model had correctly classified most of the majority of images as compared to the challenging cases like Black-grass vs. Loose Silky-bent. From the testing results, it was assured that our model is performing better and any misclassifications will be analyzed for future improvements.

This verification process was important in ensuring robustness and accuracy on the model performance, especially on an imbalanced dataset.

6 Drone Simulation for Competition

As part of the competition, we designed a simulation environment to evaluate our model's performance in real-world agricultural applications. To achieve this, we created a **30 × 30 grid-based competition field**, stored in a CSV file. This CSV file contained:

- The **image IDs** corresponding to different grid positions.
- The **classification labels** for each image, indicating whether it was a **plant** or a **weed**.

6.1 Simulating Drone Operations

To simulate autonomous agricultural drones, we structured our approach as follows:

- **Drone 1 - Classification Drone:** The first drone's role was to fly over the field and analyze the crops. It used our trained model to classify each grid cell, identifying which areas contained plants and which contained weeds.
- **Drone 2 - Weed Removal Drone:** Based on the classifications provided by the first drone, the second drone was responsible for removing weeds from the field. It navigated through the grid and targeted specific areas for weed elimination.
- **Drone 3 - Watering Drone:** The third drone focused on **watering the plants**. After weed removal, it ensured that only the correctly classified plant regions received water, optimizing resource usage.

6.2 Running the Model on the CSV Grid

To make the simulation as efficient as possible, we processed the **30 × 30 CSV grid** by iterating through each entry and running the trained model for classification. The workflow involved:

1. Extracting the **image ID** from the CSV.
2. Running the model on the corresponding image to classify it.
3. Using the classification output to simulate drone actions (classification, weed removal, and watering).
4. Storing the final results in an updated CSV file.

6.3 Storing Results in CSV Files

The output of the drone simulation was systematically recorded in CSV files to ensure transparency and reproducibility. The results stored included:

- The final classification for each image.
- The actions taken by each drone (e.g., weed removal or watering).
- A timestamp for when each operation was performed.

These CSV files provided a structured format for analyzing the efficiency and accuracy of our simulation. By cross-referencing the predicted classifications with actual labels, we were able to validate the performance of our model and refine our approach further.

7 Winning Strategy

By analyzing the dataset and observing which areas the model was consistently having difficulty with. Among the most basic of these was telling **black grass** from **sugar beet**, as it easily confused these two classes. To remedy that, we increased the number of images of black grass in the dataset, allowing it to learn the difference between black grass and sugar beets more accurately.

Some of these classes in the dataset were at the same time underrepresented, thus imbalanced. In order to take advantage of that, we included extra images for those underrepresented classes. This made our model much more robust and at the same time provided an extra barrier for competing models relying on the original distribution within the dataset. There was a non-uniform distribution in such a strategic enhancement of the dataset that led to very good improvements of our model in this competition.

8 Conclusion

Thus, the proposed system implements a precision farming system using deep learning techniques applied on autonomous drone simulations. Utilizing Fastai and ResNet50, we are able to provide a model classifying plant species accurately and further able to classify crops from weeds. Fine-tuning and further augmentation of the dataset improved performance significantly, especially on complex classifications like blackgrass vs. sugar beet.

To validate our approach, we designed a 30 × 30 grid-based competition field and simulated three autonomous drones performing key agricultural tasks, such as crop classification, weed removal, and irrigation. Results were systematically recorded in CSV files for analysis of model accuracy and efficiency.

Our winning strategy took advantage of the imbalances of the dataset by adding more images of the underrepresented classes, which ensured the classes' competitiveness. This strengthened not only our model but also competitors that depended on the standard datasets.

In conclusion, our work represents a step where machine learning, autonomous drones, and precision farming combine to optimize resource usage, minimize related costs, and improve agricultural productivity. Such next steps may include real-time field deployments, integrating additional sensor data, or employing even more sophisticated deep learning architectures to further refine weed detection and crop monitoring.

9 Declaration of Originality

I, ..., herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

[illegible] Lippstadt 01/24/2025 Ammar, Umer, and Shafin

Date&Place - Full Name

Bibliography

[Ba15] Bahr, Claudia; Forristal, Dermot; Fountas, Spyros; Gil, Emilio; Grenier, Gilbert; Hoerfarter, Rita; Jonsson, Anders; Jung, Andras; Kempenaar, Corne; Lokhorst, Kees et al.: EIP-AGRI Focus Group: Precision farming, 2015.

[He16] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian: Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[Ra19] Razaak, Manzoor; Kerdegari, Hamideh; Davies, Eleanor; Abozariba, Raouf; Broadbent, Matthew; Mason, Katy; Argyriou, Vasileios; Remagnino, Paolo: An integrated precision farming application based on 5G, UAV and deep learning technologies. In: International Conference on Computer Analysis of Images and Patterns. Springer, pp. 109–119, 2019.

[Sm17] Smith, Leslie N: Cyclical learning rates for training neural networks. IEEE Winter Conference on Applications of Computer Vision, 2017.

[UNSA19] United Nations, Department of Economic; Social Affairs, Population Division: World Population Prospects 2019, 2019.